What Software Evolution and Biological Evolution Don't Have in Common†

Chrystopher L. Nehaniv Jill Hewitt Bruce Christianson Paul Wernick
School of Computer Science
University of Hertfordshire
College Lane, Hatfield, Herts AL10 9AB, United Kingdom
{C.L.Nehaniv, J.A.Hewitt, B.Christianson, P.D.Wernick}@herts.ac.uk

***Abstract*— Understanding software change as an evolutionary process analogous to biological evolution is an increasingly popular approach to software evolvability but requires some caution. Issues of evolvability make sense not only for biological and evolutionary computation systems, but also in the realms of artifacts, culture, and software systems. Persistence through time with variation (while possibly spreading) is an analogue to variation (with heritability). Thus discrete individual replicators are not strictly necessary for an evolutionary dynamic to take place. Studying identified properties that give biological and artifact evolution the capacity to produce complex adaptive variation could shed light on how to enhance the evolvability of software systems in general and of evolutionary computation in particular. Evolution and evolvability can be compared in different domains.**

But the evolution of software systems is also very unlike that of biological entities whose existence, persistence, development, and integrity as single individuals is actively maintained by the activity of the entities themselves over a long evolutionary history. Integrity of software systems – i.e. the assumption that they are well-defined, coherent individuals that develop – is presupposed by nearly all software process approaches and limits their effectiveness. Understanding the long-term evolvability of software systems as they undergo "descent with modification" thus requires much more than a traditional Darwinian approach. We compile and discuss differences and similarities between software evolution and other instances evolution toward this end.

I. Introduction

In this paper[1] we aim to explore the connections between software evolution and biological evolution (and also evolutionary computation). All three share obvious common features in Darwin's characterization of evolution as "descent with modification", but less obvious analogies and parallels also occur (Conrad 1990, Wagner & Altenberg 1996, Ackley 2000, Nehaniv 2000, Nehaniv 2002, van Belle & Ackley 2002, Nehaniv 2003). These include notions such as modularity, sensitivity to changing requirements, as well as issues of context, and control of and types of variability that have for the most part been studied independently by workers in software engineering, evolutionary computation, and evolutionary biology.

There is growing awareness from academia, industry, and research communities of the importance of *evolvability*, tentatively defined as *the capacity to vary robustly and adaptively over time or generations in digital and natural systems.* A dialogue is beginning to emerge between various workers in areas that might benefit from a possible common framework addressing software engineering as well as biological and evolutionary computation concerns.

Darwinian evolution (characterized by heritable variation and selection) is not by itself sufficient to account for the capacity to vary and inherent phenotypic expressions of fitness.[2] Darwinian evolution is defined as a *population process* in which a population of entities ("individuals") of any sort whatsoever (organic or not), undergoes variation with differing reproductive success of individuals based at least in part on heritable characteristics (cf. [Maynard Smith and Szathmáry 1995]). Many instances of Darwinian evolutionary processes can and do occur, some of them man-made.[3]

How inheritable information determines, or more generally, contributes to other properties of the entity is referred to as the *genotype-phenotype mapping*.[4] Rigidity of genotype-phenotype mappings, as often seen in artificial examples of evolution (such as evolutionary computation), constrains the dynamics of evolution to a small space of possible biological or artificial systems. Open-ended evolution is not possible under such constraints. In biology, evolution by itself, does not fully explain the advent of genetic systems, the flexible genotype-phenotype mappings, nor heritable fitness (Nehaniv 2003). This presents a challenge both to biologists seeking to understand the capacity of life to evolve and to computer scientists who seek to harness biological-like robustness and openness in the evolution of artificial systems.

In software evolution, there is no obvious clear analogue of the *individual*. Should individuals be defined as source code programs, installations of particular software systems,

[1] † This paper extends and supersedes (Nehaniv 2000, 2002).

[2] Here and below *variability* refers to the type of change that individuals may undergo from generation to generation.

[3] Caveat: Darwinian evolution does **not** exclude the inheritance of acquired traits — indeed, such there is no reason not use such types of (so-called Lamarckian or, more correctly, non-Weismannian) inheritance for artificial evolutionary systems if it proves effective to do so. Moreover, this may well be a natural type of inheritance in considerations of software evolution (cf. Wernick 2002).

[4] *Genotype* is the inheritable information carried by an individual, while its *phenotype* is comprised by all its other properties (including form and behavior).

Second International IEEE Workshop on Software Evolvability (SE'06)
0-7695-2698-5/06 $20.00 © 2006 IEEE

software releases, organizations that produce software, or in some other way? As a consequence, it is difficult to identify any population process consisting of such individuals, or to identify heritable characteristics, or to define differential reproductive success. Similarly, what constitutes a "species"? If it is ability to interbreed, what could this mean for software? – can Windows 98 interbreed with Windows 2000?

Evolvability in biology has been variously defined as the "ability to produce adaptive variants when acted on by the genetic system" (Wagner & Altenberg, 1996), as the "capacity to generate heritable phenotypic variation" (Kirschner & Gerhart, 1998); and as characterized by 'evolutionary watersheds' opening the "floodgates to future evolution", such as segmentation and body plans (Dawkins, 1987). On the other hand, unconstrained or inappropriately constrained variability can lead to lack of stability, "cancer" (uncontrolled growth at lower levels), non-heritability of fitness, lack of evolutionary power, and so on.

In software evolution, since at least the work of Parnas (1972) and Dijkstra (1968), issues that impact evolvability have been identified in the design of software systems (e.g. structural decomposition, information hiding, modularity, requirements change).

Evolutionary computation has come a long way since Friedberg (1959) documented the difficulty of introducing random variation into conventional computer code. It is now possible to evolve programs to carry out some simple tasks (and some not-so-simple ones) by introducing appropriately constrained variability combined with robustness in the face of unfavorable variations (e.g. Koza 1992).

Software evolution and maintenance present quite different modes of variability and of descent with modification than in biological or evolutionary computation. In this realm, robustness to change in context of use (e.g. Goguen 1994) and inertia of legacy-driven concepts of "the system" (Loomes & Nehaniv 2001) are key concerns, as well as the lack of well-defined individuals (Nehaniv 2000, Loomes et al. 2005). Similar ideas such as (appropriate) modularity, duplication and subsequent divergence of functionality (cf. Ohno 1970), redundancy, phenotypic plasticity, trade-offs between freedom at different levels (e.g. Michod & Roze 1999), and the forbidding of undesirable variability while allowing potentially useful variability are common threads in evolvability issues across domains. What is the "right" level of abstraction in considering these issues for software systems?

We discuss these similarities and differences between software and other evolutionary domains, and address in what sense it is possible to transcend these disciplines in the study and application of evolvability, with particular attention to the special characteristics of software evolution phenomena.

II. Evolution of Artifacts and Software

In the realm of culture, "imitation broadly construed" serves as the replication mechanism for an evolutionary dynamic acting on so-called 'memes' (Dawkins 1976) which are somewhat ill-defined entities which comprise the "replicators" in culture, including fashion, behaviors such as ways of making a baskets or food-preparation methods, techniques and technologies, tools and other artifacts. Social learning and imitation can play the role of replication supporting a kind of heritability in human and animal cultures (Dawkins 1976, Bonner 1980). The capacity for generating adaptive variation, i.e. evolvability, in the realms of design and culture is evidently very high and seems to be supporting open-ended evolution.

A. Evolution of Artifacts

Human artifacts including objects we use daily, such as forks, knives, and paper clips are often said to evolve over many generations of design and use, in an evolutionary dynamic in which "form follows failure" - i.e. designers are inspired by their dissatisfaction with a type of artifact motivating them to make a new one (Petroski 1992). Other artifacts, including computer software, persist over time in niches with changing requirements. The change in circumstances and context of their use results in the next "generation" of artifacts having changed form - whether these are new artifacts based on the old ones (e.g. a new kind of paper clip) or old artifacts that have been modified (a new version of existing software).

Whether these artifacts persist or cease to be used depends on a kind of selection for functional and non-functional properties. For example, legacy software systems may continue to be used because they do essential work and are too expensive to understand or replace. A model of car may be successful because of fuel efficiency or because it looks fashionably impressive and attracts consumers. This kind of heritability analogue (persistence of characters over generations of artifacts) makes selectable entities of tools, or of the knowledge of how to make the tools. 'Tools' here includes non-physical tools such as software development methodologies, theories of evolution, the decimal expansion, methods of calculation, algorithms, etc. (Nehaniv 1997). Persistence of 'species' of artifacts and memes in particular economic-ecological niches combined with variability in their design provides for what appears to be evolution without [explicit] genetics.

Just as biological individuality has arisen in the origin of eukaryotes (cells having a nucleus) and of multicellular lifeforms from the combination of previously reproductively independent entities, new artifacts and software arise from the combination of existing components into new artifacts. Thus persistence with variation and selection (and many other phenomena seen in organismal evolution) are also present in the world of artifacts providing a dynamic which we might perhaps not too metaphorically identify with the heritability, variation and selection of Darwinian evolution.

B. Tools and Cognition

Tools will tend to be used in accidental ways not anticipated by their designers. Tool design will change as a result of human needs and desires, and tools will change the humans who use them. Designs that can support requirements change while continuing to meet human needs will be more "evolutionarily pregnant" (to use Dawkins' phrase) - i.e. more evolvable.

Evolvable systems are "easy to sellotape", i.e. easy to adapt and modify in the face of external changes and unforeseen circumstances (cf. Gatlin 1972).

Ever since humans (and their ancestors) started extending their capabilities with tools, we have been "cyborgs" - organisms whose embodiment is modified and extended via technologies (Haraway 1991, Nehaniv 1997, Clark 2003). Tools change what we are and how we interact with our world, e.g. spoons, drums, clothing, eye glasses, money, language, writing, arithmetic, etc. Designers externalize ideas for magnifying human capacity through the artifacts they create. Conversely, the design and use of artifacts often impacts human cognition in many unforeseen ways. The tools and artifacts that humans design change our interaction with the world around us. Evolutionary pathologies (in biology or other evolutionary systems) are the persistence or accumulation of undesirable features that may lead to disaster in the long-term, simply as a consequence of evolutionary dynamics – cf. the notion of "tragedy of the commons" and other examples from [Altenberg 2000]. In order to avoid evolutionary pathology in technological evolution it is useful to ask when designing any new artifact: How will it serve the interests of people? How will it enhance human cognitive and social capabilities? What will be the technologically induced adaptations of humans to the artifact? Does it, and, if so, how does it integrate technological and human processes while respecting human wholeness? Issues of ethics and of empowerment arise with the introduction of new artifacts. How will this tool affect human cognition and behavior? How will using and interacting with this technology change who we are? Will it, and, if so, how will it help optimize the relationship between the "CYB" and "ORG" (Mey 1997, Gorayska & Mey 1996, Dautenhahn & Nehaniv 2000). Tools "configure" their users: the users of tools must adapt to themselves to their tools. What kind of cognitive calluses might the use of an artifact engender? (Nehaniv 1999). If a tool proliferates, what will be the impact on the environment? And, pragmatically, to what extent are questions such as these submerged in the face of technology-driven and economic demands or imperatives?

C. Evolution of Software

In software engineering, change in requirements and context of use is the major factor in cost and impacts the areas of requirements engineering, software maintenance, and software evolution. Evolvability as a capacity to generate adaptive variation in tandem with continued persistence of software artifacts would be welcome in software engineering. Factors supporting evolvability in artifact and software design, systems theory, and digital evolution have analogues in biological evolvability. Certain properties of artifacts and of software are recognized as enhancing their capacity to support and be adapted to changing requirements and context of use, yielding flexibility of use and variability tolerance.

Software maintenance - the process of changing a software system after its release - is the most expensive part of software costs, estimated as accounting for as much as 80% or more of effort (e.g. Sommerville 1996, ch. 22; Pressman 1992, ch. 20; Lehman 2005). Corrective, adaptive, and perfective maintenance of software to satisfy mutable requirements provides a mechanism for inheritance and (highly non-random) variability of software artifacts. Requirements, economic and power relationships, accountability, and usefulness of the software in functioning in context all contribute to the dynamic and social process of software requirements engineering as the reconciliation of technical and social issues (Goguen 1994, 1996). Brittleness of current software systems in the face of requirements change is thus a huge problem for software engineering. Existing software is not robust against changing requirements and contexts of use.

Re-use (not replication), modularity, information hiding, encapsulation, and object-oriented 'inheritance' are some mechanisms that software engineers have developed in order to provide robustness to environmental and requirements change in the course of software evolution. These are intended to improve the capacity of software for adaptation. Nevertheless, practical methodologies and theoretical understanding of how to build maintainable software are still for the most part wide open research areas. We still do not have really good methods for how to grow or evolve software systems.

III. Evolvability in Biology and Evolutionary Computation

Recently it has grown increasingly clear that evolutionary biology still lacks a complete account of the capacity of populations of living systems to change their architectures and increase in complexity in the course of evolution. We have only very limited knowledge of the origins and evolution of genotypes and of genotype-phenotype relations. Whereas most population geneticists start with the assumption of a fixed -often abstract- genetic system, experience in evolutionary computation shows that the choice of genetic representation and the mechanisms for variability crucially affect the capacity of the evolution to produce non-lethal phenotypic adaptations (Altenberg 1995, Wagner & Altenberg 1996, Kirschner & Gerhart 1998).

Thus the evolvability of life on earth - the capacity to produce complex adaptations, such has new metabolic pathways, organs, and body plans - is not really explained by merely saying that it must "somehow" be a result of Darwinian evolution (since many artificial instances of Darwinian evolution lead to no such adaptations). Evolution, at least in evolutionary computation, obviously fails even on many simple optimization tasks if the genotype-phenotype relationship is brittle, e.g. when variability is produced by varying bits or characters in standard computer programs (Friedberg 1959).[5] In contrast to such brittleness in the face of variation, when variation is less likely to be lethal and more likely to have a non-negligible chance of producing useful adaptation, even populations of computer code entities can exhibit complex

[5] It is interesting to note that similar techniques are sometimes now used to do coverage analysis for assertion violation as an application of variability to debugging.

evolutionary adaptations, as shown by the examples of genetic programming (Koza 1992) and self-replicating programs in Tierra (Ray 1992). Populations of individuals with robust 'encoding' are more evolvable (cf. Gatlin 1972), and artificial examples have produced human-competitive applications in the design of electronic circuits (Koza & Bennett 1999, Koza 2003) and aeronautics (Bannasch 2001).

IV. Individuality, Persistence and the Heritability of Fitness

A. Evolvability without Explicit Genetics

Certain classes of artifacts persist either through successive generations of design, or merely as (possibly modified) single instances as they continue to be used. The context and requirements of use change with time in ways that cannot be foreseen - during initial design and development, and also after deployment. Software systems go through various versions and releases, and software components are combined and re-used in new ways. Behaviors, technologies, and ideas are transmitted between individuals, often with variation and persistence over long periods. Without a genetic system, could there be an evolutionary dynamic present in these examples? If so, how is it instantiated and in which cases do we see a significant capacity to evolve complex adaptations? What factors support this kind of evolvability?

B. Mergers of Selectable Entities

Biological organisms reproduce and thus form readily identifiable entities comprising populations in which the ingredients of evolution -heritability, variation, selection- are found. During major evolutionary transitions such as the origin of differentiated multicellular organisms, in which new levels of individuality arise (comprised of pre-existing smaller replicators), what is meant by "fitness", and questions of its heritability at various levels are complex issues (Buss 1987, Michod & Roze 1999). Another example of new selectable entities arises in symbiogenesis, i.e. the advent of new species or higher level individuals from mergers of organisms living in close proximity to one another (e.g. one inside the other). For instance, "gardens of bacteria" whose reproductive fates became inextricably intertwined (Margulis 1983) gave rise via symbiogenesis to the first eukaryotic cells, having components including a nucleus, numerous energy 'powerhouses' (mitochondria), and other components (organelles) that were derived from simpler, previously free-living, bacterial ancestors. At intermediate stages of such transitions, replication at the higher level is uncertain; fitness at a new level has not yet congealed and is still becoming heritable.

In the case of software, re-use and combining of modules, code, or programs together into more complex systems is reminiscent of such evolutionary mergers. The merged structures can then in turn themselves become the focus of further software evolution.

C. Growth by Accretion

Large software systems' growth is often characterized by accumulation of layers of legacy, gradual addition of small parts and changes, and of re-use of structure without much regard for what underlies it. Such growth by accretion suggests that in some ways software change may be more like coral reef growth than, say, bacterial evolution. From this point of view, software evolutionary processes are far from Darwinian evolution and more like the growth of a colony of organisms that interacts with and transforms its environment (cf. [Jackson, Buss, and Cook 1986]).

D. Persistence as Weak Heritability

A new level of individuality implies a niche for the new individuals to make their living in. Even before individual fitness at the new level becomes heritable, entities occupying the niche have an obvious property: they persist in the niche. The persistence of entities through time is a property of a population evolving in its environment and (possibly changing) niche.[6]

Natural selection acts to determine whether an individual's properties will persist. Thus although successful characteristics of an individual may certainly be inherited by its offspring, survival of the individual itself already entails the persistence of that individual's properties. Persistence is therefore a (weaker) analogue of heritability. Can one make sense of evolution in the absence of heritability? Yes, persistence in a niche - possibly with variability and with spread over time - plays the same role as heritability in natural extensions of the Darwinian paradigm (Nehaniv 2000). This new view of persistence makes it possible to extend this paradigm more generally into the realm of artifacts and design. Without our necessarily being able to identify any discrete, self-reproducing entities, persistence through time while possibly growing and spreading serves the same role as heritability does in a classical Darwinian paradigm.

A degenerate case of this is persistence without change, growth, or variation - e.g. of a stone existing without substantial change over a long period of geological time. A less degenerate case is growth and spread without variability, e.g. in the growth of crystals. Persistence with growth and variation is apparent in the lifespan of single living things, maintained software systems, coral reefs, cities, and many other entities; within these cases we have persistence and variability providing analogues of heritability without strict reproduction of individuals. Design and cultural traditions, and generations of software releases, provide examples closer to biological evolution acting on populations but still lack well-defined self-reproducing individuals. Looking at these as examples along a continuum from persistence of entities to reproduction of self-reproducing individuals generalizes the notion of Darwinian evolution to many other realms having

[6]These considerations of persistence in a niche are related to but not directly dependent on the notion of 'species'. A species (in addition to having other characteristic properties) consists of a population of individuals whose lineages persist within a common niche.

many similarities. In particular, in each case one can ask about the capacity of the generalized evolutionary system to produce adaptive variation. That is, one can study evolvability phenomena and the factors that support them in these systems. It turns out that many non-biological systems showing the capacity for generating adaptive variation share many features with biological ones.

V. Properties of Evolvable Systems

For artifacts (including software) we can define *evolvability* as *the capacity of the systems, organizations and networks producing them to give rise to adaptive variants that flexibly meet changing requirements over the course of long-term change*.

Many of the properties thought to enhance software evolvability are strongly analogous to properties considered to be of great importance in biological evolvability and evolvability in evolutionary computation. A very incomplete list of (non-orthogonal) factors includes (see cited references for more details):

1) Modularity - low interdependence ('pleiotropy') between functionally distinct components, correspondence of units of coding with units of function (biology: Conrad, 1990, Wagner & Altenberg 1996; software: information hiding and encapsulation (Parnas 1972); cf. functional decomposition, structured programming (Dijkstra 1968), and "object-orientation"); weak linkage; compartmentation.
2) Facilitation of extra-dimensional bypass (Conrad 1990), e.g. via duplication and divergence (Ohno 1970, Nehaniv & Rhodes 2000): the creation of new routes for evolutionary change by adding new dimensions for potential adaptive variation.
3) Robustness to genetic variability (Conrad 1990, Ray 1992, Koza 1992)
4) Phenotypic robustness, developmental tolerance and constraints, embryologies (Conrad 1990; Gerhart & Kirschner 1997; Kirschner & Gerhart 1998; West-Eberhard 1998; Maynard Smith et al. 1985); Baldwin's effect (the internalization, over generations, of adaptation to external evolutionary pressures).
5) Redundancy: belt-and-suspenders phenomena; duplication of components (von Neumann 1956).
6) Switches - use-retargetable mechanisms which various different kinds of signals could be configured to control (via 'signal transduction'); re-use; genetic regulatory networks, transcriptional control; homeotic genes and developmental cascades; hierarchical organization (Simon 1969); informational patterning; positional systems in development (L. Wolpert); interoperability.
7) Conservation of Core Mechanisms, Diversification of Regulatory Mechanisms; customization and re-use.
8) Robustness to environmental and context change - fault-tolerance, requirements engineering, defensive programming: results in (1) reduced lethality due to mutations or varied environmental conditions; (2) support of phenotypically useful traits without 'genetic' change, modulated to provide (3) control of kind and amount of phenotypic variation produced to achieve a functional state regardless of initial configuration and perturbation of external factors.
9) Search behavior in biological systems: exploratory epigenetic mechanisms for variation and selection - harnessing evolutionary dynamics: within an individual (Gerhart & Kirschner 1997) - e.g. the formation (according to Darwinian dynamics) of temporary scaffolding structures (via microtubule synthesis) in cytoskeleton (cell skeleton), in cell motility, and in reproductive cell functions (meiosis and mitosis); in self-organizing processes in ant foraging behavior; in neural and vascular growth and morphogenesis; in evolution within an immune system; as well as learning, adaptation, and evolution (Holland 1975).
10) Genotype-Phenotype relations or variability generation mechanisms under evolutionary control.
Selection for robustness can have as a non-selected by-product the following properties which enhance evolvability: (1) phenotypic variation becomes tolerated and possible; (2) phenotypic variability becomes heritable, since similar genes in similar environment yield similar development; and (3) developmental versatility leads to increased phenotypic variability serving as fodder for the "next round of evolution".

VI. Evolvability of Software

A. Presupposition of Individual Integrity

A serious problem, in our view, of most models of software development and evolution is their blind acceptance of the "life cycle" metaphor that tacitly presupposes the integrity of a software system as a single coherent individual which will "develop" properly from an "embryonic" requirements specification to a "mature" software system if properly cared for (Lam and Loomes 1998, Loomes and Jones 1998, Loomes and Nehaniv 2001, Loomes et al. 2005). However, this viewpoint is dangerous if "the system" does not refer to any well-defined entity (e.g. before requirements are discussed), or if it refers to a multitude of vaguely defined entities of questionable ontological status (given e.g. by user expectations, specification documents, etc.) which may or may not have any existence as a working "system", or even to a multitude of deployed systems under the same product name (such as 'Microsoft Windows') – see (Loomes et al. 2005) for a detailed discussion of this.

This situation in software development and evolution is radically unlike the case in biological evolution, where individual growing and developing organisms actively acquire and metabolize the resources necessary to construct and maintain their own persistent individual bodies and integrity in the face of harsh entropic challenges from the surrounding universe. Software systems in contrast generally take no active role in their maintenance, but persist due to inertia or active work on the part of human communities of actors. Nevertheless, stakeholders and organizations committed to a developing

system can bring about massive absorptions of resources into attempts to promote a system's realization and integrity – cf. (Latour 1987, Wernick et al. - this volume).

Given the considerations on lack of clear individuals in software evolution discussed here, it is abundantly clear that the existence and persistence of single entity over a longer temporal extent is no longer to be taken for granted when we move from the world of organisms or everyday objects into the world of software. It might even be the case that the notion of a developing individual – "the system" is not an appropriate metaphor in the realm of software maintenance and evolvability.

B. Evolution, Naming and Persistence Entities

Naming of objects and natural phenomena – whether individuals or classes of them – works in a particular way that generally pre-supposes the existence, integrity, and the continued persistence of the entities named. When we name persons or animals these assumptions are naturally satisfied, and we are able to give a name without much ambiguity to a person even the individual changes and develops through radically different forms in the course of life [Loomes et al. 2005]. Biological organisms have a natural persistence and integrity that allows us to do this. Our usual ways of using giving things names have been co-opted (or "exapted", i.e. re-used and adapted for unanticipated applications over the course of time) to a new realm when we apply naming to software systems. Meanwhile, software systems and their often dynamically changing identities and characteristics emerge from and are transformed by the activity of networks of actors working in contexts of internal, external, technological and social constraints and requirements.

In evolutionary computation or in biological systems, there is generally a good measure of agreement on what constitutes an individual in an evolving population. In software evolution, such agreement is markedly absent. (In cases where it seems to exist, it is the result of tacit acceptance of an arbitrary norm, e.g. in version numbering, rather than logical necessity or rational inquiry.)

C. Individuals – Units of Selection

Darwin's broad sense of evolution in organismal species as "descent with modification" applies at the level of populations of the individuals in a species over time undergoing a dynamical process with heritability, variability, selection ("struggle for existence") and limited resources. Populations rather than individuals evolve (although individuals may change and develop in their life times (as, for example, in the life cycle of a butterfly), but *well-defined individuals* are required for the Darwinian theory to apply [Maynard Smith and Szathmáry 1995], [Buss 1987], [Michod 1999]. Persistence of changing entities becomes a weaker analogue of individuals in an evolutionary dynamic and occurs also for other candidate spheres of evolutionary phenomena such as memes, software, or physical technological artifacts ([Nehaniv 2000] and above).

The software engineering community uses the term 'evolution' in a broader sense that also focuses on the descent with modification of software systems, but does not actually presuppose populations of competing individuals of the same species. Competition is instead for a given niche, and the makers of new software products may seek to invade, create or expand software niches.

Where software systems are seen as being modified and maintained in the face of changing requirements and contexts of use [Goguen 1994], [Lam and Loomes 1998], selection is not usually discussed, but explicit empirical laws for the evolution of particular classes of software systems can be formulated [Lehman 1980].

D. Genes, Species, Individuals

Even more importantly, software evolution differs from biological evolution in a fundamental manner: There is currently no well-circumscribed notion of what constitutes an individual software system, nor of heritable material, nor of species. The lack of clear-cut software individuals is a serious obstruction to the use of Darwinian evolutionary analogues for software. Related to this is the fact that one has no clear analogue of 'gene', the unit of heritable material. Genetics in biology does not completely determine or 'specify' an individual, but constrains its potentialities for developing in interaction with its environment. Genes are what make these potentialities heritable and subject to natural selection. In software evolution, it might be otherwise, e.g. program code may determine software behavior completely; but for software, no one has as yet produced a compelling answer to the question "what is [or should be] a gene?"

Biological species are often defined by the capacity to interbreed, and by isolation or barriers to breeding outside the group. For software, what should constitute a species has no obvious clear answer. Software does reproduce by interbreeding: incremental releases are in some sense uniparental, and systems that combine multiple software components can not be usefully said to be of the same species as their components.[7]

Could we see the customization of a generic software product via parameters and installation options as phenotype variation? Would this put the generic product in the role of species and the individual copy as individual? Customization could also form the basis of ideas for changes in future releases. Perhaps a 'system as fielded' could be considered an individual, and its lines of code or its constituent modules might be considered as 'genes' (potentially inheritable – reuseable – in other programs). However, all these suggestions are highly debatable, and many alternative, equally plausible, but conflicting interpretations of gene, species, and individual in the realm of software seem possible.

[7]This last example seems to be closer to an instance of symbiogenesis, the creation of new species by mergers of old ones – see above and [Margulis and Sagan 2003].

VII. Conclusions

Modularity, genotype-phenotype relations, the ability to replicate, heritability of fitness, and evolvability are derived states of biological systems that have scientific explanations (some currently still being worked out). Integrity of coherent individuals cannot be presupposed for software evolution. Persistence over time with variational change can play the role of heritability with variation in a Darwinian (but not necessarily Weismannian evolutionary dynamic). From this point of view, evolution and evolvability of artifacts and software systems, designs, culture and memes can be compared. Studying the properties that make such evolution possible, and that make biological and cultural systems exhibit evolvability, shed useful light on ways to improve evolutionary computation and the design and evolution of software systems in general. This study also reveals a cross-disciplinary unity at a high level of the mechanisms of evolvability in different realms including biology, artifacts, culture, and software systems, but with some very important differences that cannot be ignored.

Software evolution is quite different from biological evolution or instances of evolution in artificial life and evolutionary computation. This is due in large part to the lack of an adequate analogue of individual. Software systems at present do not have 'inherent coherence': unlike biological individuals they do not at present engage actively in their own production, self-maintenance, and adaptation to the environment. The dynamics of reification, persistence, and continuity of software "individuals", as well as an explicit, defensible definition of the term (or, alternatively, of an analogous concept replacing the notion of "individual"), must be addressed by a successful theory of software evolution.

References

[Ackley 2000] Ackley, D.H. "Real artificial life: Where we may be". In M.A.Bedau, J.S.McCaskill, N.H.Packard, and S.Rasmussen (eds.) *Artificial Life VII (Proceedings of the Seventh International Conference on Artificial Life)*, Cambridge, MA: The MIT Press (A Bradford Book) (2000).

[Altenberg 1995] Altenberg, L. 1995, Altenberg, L. 1995, Genome growth and the evolution of the genotype-phenotype map. In W. Banzhaf and F. H. Eeckman, eds., *Evolution as a Computational Process*, Springer Verlag, pp. 205-259.

[Altenberg 2000] L. Altenberg, Evolvability Checkpoints Against Evolutionary Pathologies, Invited paper for the Evolvability Workshop at the Artificial Life 7 Conference, Portland Oregon. `http://homepages.feis.herts.ac.uk/~comqcln//al7ev/cnts.html`

[Bannasch 2001] Bannasch, R. (2001) From soaring and flapping bird flight to innovative wing and propeller constructions. Progress in Astronautics and Aeronautics 195:453-471.

[Bonner 1980] Bonner, J. T. (1980) The Evolution of Culture in Animals, Princeton.

[Buss 1987] L. W. Buss (1987). *The Evolution of Individuality*, Princeton University Press.

[Clark 2003] A. Clark, *Natural-Born Cyborgs*. Oxford University Press, 2003.

[Conrad 1980] Conrad, M. 1990, The geometry of evolution. Biosystems 24:61-81.

[Dautenhahn and Nehaniv 2000] Dautenhahn, K. and Nehaniv, C. L. 2000, Living with Socially Intelligent Agents: A Cognitive Technology View, in *Human Cognition and Social Agent Technology*, K. Dautenhahn, ed., John Benjamins, pp. 415-426.

[Dawkins 1976] Dawkins, R. 1976. *The Selfish Gene*, Oxford University Press.

[Dawkins 1989] Dawkins, R. 1989, The Evolution of Evolvability. In: Artificial Life, C. Langton, ed. Addison-Wesley.

[Dijkstra 1968] Dijkstra, E. W. 1968, Goto statement considered harmful, Communications of the Association for Computing Machinery, 11(3):147-148.

[Friedberg 1959] Friedberg, R. M. 1959, A learning machine, IBM J. Res. Dev., Part II, 3:181-191.

[Gatlin 1972] Gatlin, L. L. 1972, Information Theory and the Living System, Columbia University Press.

[Gerhart and Kirscher 1997] Gerhart, J. and Kirschner, M. 1997, Cells, Embryos, and Evolution: Towards a Cellular and Developmental Understanding of Phenotypic Variation and Evolutionary Adaptability, Blackwell Science.

[Goguen 1994] Goguen, J. 1994, Requirements Engineering as the Reconciliation of Technical and Social Issues, in *Requirements Engineering: Social and Technical Issues*, edited Marina Jirotka and Joseph Goguen, Academic Press, 1994, pp. 165-199.

[Goguen 1996] Goguen, J. 1996, Formality and Informality in Requirements Engineering, Proceedings, Fourth International Conference on Requirements Engineering, IEEE Computer Society, April 1996, 102-108.

[Gorayska and Mey 1996] Goryaska, B. and Mey, J. L. 1996, Of Minds and Men, in Cognitive Technology: IN Search of a Human Interface (B. Gorayska and J. L. Mey, eds., Advances in Psychology, vol. 113) Elsevier/North Holland, pp. 27-39.

[Haraway 1991] Haraway, D. 1991, A Cyborg Manifesto: Science, Technology, and Socialist-Feminism in the Late Twentieth Century. In D. Haraway, Simians, Cyborgs and Women: The Reinvention of Nature, Routledge, pp. 149-181.

[Holland 1975] Holland, J. 1975, Adaptation in Natural and Artificial Systems, MIT Press.

[Jackson, Buss, and Cook 1986] Jackson, J. B. C., Buss, L. W. and Cook, R. E. 1986. Population Biology and Evolution of Clonal Organisms. Yale University Press.

[Kirschner and Gerhart 1998] Kirschner, M. & Gerhart, J. 1998, Evolvability, Proc. Natl. Acad. Sci. USA, Vol. 95, pp. 8420-8427, July.

[Koza 1992] Koza, J. R. 1992, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press.

[Koza 2003] Koza, J. R. 2003. Human-competitive applications of genetic programming, Advances in evolutionary computing: theory and applications, Springer-Verlag.

[Koza and Bennett 1999] Koza, J. R. and F. H. Bennett III, 1999, Automatic synthesis, placement, and routing of electrical circuits by means of genetic programming. In: Advances in Genetic Programming, vol. 3, MIT Press, pp. 105–134.

[Lam and Loomes 1998] W. Lam and M. J. Loomes (1998). Requirements Evolution in the Midst of Environmental Change: A Managed Approach, *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering (CSMR'98)*, IEE Press, pp. 121–127.

[Latour 1987] B. Latour (1987). *Science in Action*, Harvard University Press.

[Latour 1996] B. Latour (1996). *Aramis or the love of technology*, translated by C. Porter, Harvard University Press.

[Lehman 1980] M. M. Lehman (1980). Programs, Life Cycles and Laws of Software Evolution, *Proceedings of the IEEE* 68(9):1060–1076,

[Lehman 2005] M. M. Lehman (2005). The Role and Impact of Assumptions in Software Development Maintenance and Evolution, Proc. Intl. IEEE Workshop on Software Evolvability 2006, 3-14, IEEE Computer Society Press.

[Loomes and Jones 1998] M. J. Loomes and S. Jones (1998). Requirements Engineering: A Perspective through Theory-Building, *Proc. Third International Conference on Requirements Engineering*, IEEE Computer Society Press, pp. 100–107.

[Loomes and Nehaniv 2001] M. J. Loomes and C. L. Nehaniv (2001). Fact and Artifact: Reification and Drift in the History and Growth of Interactive Software Systems, *Proc. Fourth International Conference on Cognitive Technology: Instruments of Mind*, Springer Lecture Notes in Computer Science, vol. 2117, pp. 25-39.

[Loomes et al. 2005] M. J. Loomes, C. L. Nehaniv, P. Wernick, "The Naming of Systems and Software Evolvability", IEEE International Workshop on Software Evolvability, IEEE Computer Science Press, pp. 23-28, 2005.

[Margulis 1981] Margulis, L. 1981, Symbiosis in Cell Evolution, Freeman.

[Margulis and Sagan 2003] Lynn Margulis and Dorion Sagan (2003). *Acquiring Genomes: A Theory of the Origins of Species*, Basic Books.

[Maynard Smith and Szathmáry 1995] J. Maynard Smith and E. Szathmáry (1995). *The Major Transitions in Evolution*, W.H. Freeman.

[Mey 1997] Mey, Jacob. 1997, Personal communication, questions presented at Second International Conference on Cognitive Technology, 25 August 1997, University of Aizu, Japan.

[Michod 1999] R. E. Michod (1999). *Darwinian Dynamics: Evolutionary Transitions in Fitness and Individuality*, Princeton University Press.

[Michod and Roze 1999] Michod, R. E. and Roze, D. 1999, Cooperation and conflict in the evolution of individuality. III. Transitions in the unit of fitness. In: C. L. Nehaniv, editor, Mathematical & Computational Biology: Computational Morphogenesis, Hierarchical Complexity & Digital Evolution, Lectures on Mathematics in the Life Sciences, vol. 26, American Mathematical Society, Vol. 26: 47-91.

[Nehaniv 1997] Nehaniv, C. L. 1997, Algebraic Models for Understanding: Coordinate Systems and Cognitive Empowerment. In: Second International Conference on Cognitive Technology: Humanizing the Information Age, IEEE Computer Society Press, pp. 147-162.

[Nehaniv 1999] Nehaniv, C. L. 1999, Story-Telling and Emotion: Cognitive Technology Considerations in Networking Temporally and Affectively Grounded Minds, Third International Conference on Cognitive Technology: Networked Minds (CT'99), pp. 313-322. `http://homepages.feis.herts.ac.uk/~comqcln/nehaniv_ct99.pdf`

[Nehaniv 2000] C. L. Nehaniv (2000). Evolvability in Biological, Artifacts, and Software Systems. In: C. C. Maley and E. Boudreau (Eds.), *Artificial Life 7 Workshop Proceedings - Seventh International Conference on the Simulation and Synthesis of Living Systems*, Reed College, pp. 17-21.

[Nehaniv 2002] C. L. Nehaniv, "What Do Software Evolution and Biological Evolution Have in Common", Software Evolution and Evolutionary Computation Symposium Abstracts, EPSRC Network on Evolvability in Biological and Software Systems, pp. 1-2, 2002.

[Nehaniv 2003] C. L. Nehaniv (2003). Evolvability, *BioSystems: Journal of Biological and Information Processing Sciences* 69(2-3):77–81.

[Nehaniv and Rhodes 2000] Nehaniv, C. L. & Rhodes, J. L. 2000. The Evolution and Understanding of Biological Complexity from an Algebraic Perspective, Artificial Life, 6(1): 450-67.

[Ohno 1970] Ohno, S. 1970, Evolution by Gene Duplication, Springer Verlag.

[Parnas 1972] Parnas, D. 1972, On the criteria to be used in decomposing systems into modules, Communications of the Association for Computing Machinery, 15(2):1052-1058.

[Petroski 1992] Petroski, H. 1992. The Evolution of Useful Things, Vintage Books.

[Pressman 1993] Pressman, R. S. 1993, Software Engineering: A Practitioner's Approach, 3rd ed., McGraw-Hill. cob. 1997, Personal communication, questions presented at Second International Conference on Cognitive Technology, 25 August 1997, University of Aizu, Japan.

[Simon 1969] Simon, H. 1969, The Sciences of the Artificial, MIT Press.

[Sommerville 1996] Sommerville, I. 1996. Software Engineering, 5th ed., Addison-Wesley.

[van Belle and Ackley 2002] Van Belle, T, and Ackley, D.H. "Code Factoring and the Evolution of Evolvability". In The Proceedings of GECCO-2002 (2002, July).

[von Neumann 1956] von Neumann, J. 1956, Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components, in Automata Studies (C. E. Shannon and J. McCarthy, eds., Annals of Mathematics Studies, Number 34), Princeton, pp. 43-98.

[Wagner and Altenberg 1996] Wagner, G. P. & Altenberg, L. 1996, Complex Adaptations and the Evolution of Evolvability, *Evolution*, Vol. 50, No. 3, pp. 967-976, June.

[West-Eberhard 1998] West-Eberhard, M. J. 1998, Evolution in the light of developmental biology, and vice versa, Proc. Natl. Acad. Sci. USA 95:8417-8419.

[Wernick 2002] Wernick, P. 2002, Analogies for Software Evolution: Darwin or Lamarck? Software Evolution and Evolutionary Computation, University of Hertfordshire, 7-8 February 2002 (EPSRC Network on Evolvability in Biological and Software Systems Symposium)

[Wernick et al. 2006] Wernick, P., Hall, T., and Nehaniv, C. L., Software Evolutionary Dynamics Modelled as the Activity of an Actor-Network, Proceedings 2nd Intl. Workshop on Software Evolvability, IEEE Computer Society Press, (this volume - 2006).